Using a Space Filling Curve for the Management of Dynamic Point Cloud Data in a Relational DBMS

Stella Psomadaki
P5 Presentation

Theo Tijssen
Peter van Oosterom
Roderik Lindenbergh
Ulf Hackauf
External: Fedor Baart
Contents

- Introduction
- Methodology
- Results
- Conclusions & Future work
Introduction
What is a Point cloud?
Point clouds

• Rapid growth in point cloud usage

• The management of point clouds is challenging

• Typically managed using files (e.g. LAS, LAZ)

• …But, DBMSs provide point cloud management solutions.
Management of PC in DBMS

Current approaches:

• Oracle *SDO_PC*

• PostgreSQL *pgpointcloud*

Organise points in **blocks**, meaning groups of spatially close points

...or use a normal **flat table**

Source: Massive point clouds for eSciences
http://www.gdmc.nl:8080/mpc/
Dynamic point clouds

• Today, developments in point cloud acquisition devices allow repeated scans of the same area

• Dynamic point clouds
  - growing datasets
  - *time is an additional dimension*
Managing dynamic PC?

- Blocks
  - compact storage with better scalability, less overhead, better compression
  - overlapping blocks, adding new data not trivial
- Flat
  - flexible, insertions trivial, Use a SFC to improve the organisation (van Oosterom et. al., 2015)
  - large storage requirements, overhead
Space Filling Curves

- Apply a linear ordering to a multidimensional domain

- Why?
  - Dimensionality reduction
  - Full resolution curve
  - Clustering of points

Source: http://asgerhoedt.dk

Space Filling Curves

(a) Row order

(b) Row prime

(c) Morton or Peano

(d) Hilbert

(e) Grey

(f) Cantor - diagonal

Space Filling Curves

• Morton Curve

• Bitwise interleaving

Example:

x = 4 or 0100 in binary

y = 6 or 0110 in binary

morton = 00111000 or 56
Research Question

Is a Space Filling Curve (SFC) approach an appropriate method for integrating the space and time components of point clouds in order to support efficient management and querying (use) in a DBMS?
Methodology: A Space Filling Curve approach
Requirements

Requirements for spatio-temporal data management [Adapted from Gaede and Gunther, 1998]:

• Should support operations other than just retrieval of the data.

• Should be dynamic: support insertions

• Should be scalable: adapt to growing database.

• Should be efficient in terms of time (and space): minimise as much as possible the number of disk accesses
Important queries

- **Space** queries: all points located in a specific area over the complete time range
Important queries

- **Space - time** queries: all points located in a specific area during a specific time range
Important queries

- **Time** queries: all points of a specific time moment or range, for the whole spatial domain
A SFC approach

Structuring space and time is not a trivial problem. Contradiction:

- Points close in space and time should be stored (up to a certain extent) in contiguous blocks in disk, for fast spatio-temporal retrieval.

- Already organised points should not be reorganised when inserting new data, for fast loading.
A SFC approach

**Integrated** space and time approach: all dimensions have equal part in SFC.

Two treatments of z:

1. as an attribute.
2. as part of the SFC key.
A SFC approach

**Non-integrated** space and time approach: time dominates over space.

Two treatments of z:

1. as an attribute.

2. as part of the SFC key.
A SFC approach - Loading

Two step approach:

- **Preparation**: Read files and convert to SFC key, according to
  - integration of space and time,
  - treatment of z and
  - scaling of time

  The data are bulk loaded into a normal heap table

- **Loading**: Sort the data based on the key into an Index Organised Table (data stored in the B-Tree index)
A SFC approach - Query

- Translation of the n-D query geometry into a number of continuous runs on the curve.

- Take advantage of the quadrant recursive characteristic of Morton curve: Use a Quadtree/Octree/\(2^n\)-tree

- The maximum depth of the tree affects:
  - the number of ranges
  - the approximation of the query geometry
A SFC approach - Query

Multi-step query procedure

• Filter step: approximate query geometry using the $2^n$-tree

• Fetch the approximated data and decode back to the original dimensions

• Refinement step: Detect the false hits using a Point in Polygon operation, or time and z refinement.
A SFC approach - Query

Identify Tree Cells
Direct neighbour merging

Reduce the number of ranges without affecting the approximation, by merging neighbouring ranges.

Figure a: Original 3 ranges
Figure b: Direct neighbour merging (1 range)
Direct neighbour merging

Merge of direct neighbours
Merging to maximum number

- Impose upper limit to the number of ranges
- Approximation gets slightly worse
- More false hits fetched during the filter step
Merging to maximum number

Additional space

Original 11 ranges

maximum 2 ranges

maximum 3 ranges
Merging to maximum number

merged

Expansion

Merge to max. number (30)
Merging to maximum number

merged

Expansion

Merge to max. number (20)
A SFC approach - Query
A SFC approach - Query

- partially
- inside
A SFC approach - Query

Multi-step query procedure

- Filter step: approximate query geometry using the $2^n$-tree

- Fetch the approximated data and decode back to the original dimensions

- Refinement step: Detect the false hits using a Point in Polygon operation, or time and z refinement.
A SFC approach - Query

Multi-step query procedure

- Filter step: approximate query geometry using the $2^n$-tree

- Fetch the approximated data and decode back to the original dimensions

- Refinement step: Detect the false hits using a Point in Polygon operation, or time and z refinement.
A SFC approach - Query

Point In Polygon operation

- partially
- inside
A SFC approach - Query
Results
Benchmark design

• Measure performance of storage space, loading time and query response time

• Datasets
  - Sand Engine
  - Coastline of the NL

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Time resolution</th>
<th>Spatial resolution</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sand Engine</td>
<td>day</td>
<td>mm</td>
<td>100,000 pts/day</td>
</tr>
<tr>
<td>Coastline</td>
<td>year</td>
<td>cm</td>
<td>500 million pts/year</td>
</tr>
</tbody>
</table>
Benchmark design

- Benchmark stages

Table 1. The benchmark stages of the Sand Engine dataset

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Points</th>
<th>Days</th>
<th>Size (MB)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>18 M</td>
<td>230</td>
<td>347</td>
<td>2000 - 2002</td>
</tr>
<tr>
<td>Medium</td>
<td>44 M</td>
<td>554</td>
<td>836</td>
<td>2000 - 2006</td>
</tr>
<tr>
<td>Large</td>
<td>74 M</td>
<td>931</td>
<td>1414</td>
<td>2000 - 2015</td>
</tr>
</tbody>
</table>

Table 2. The benchmark stages of the Coastline dataset

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Points</th>
<th>Years</th>
<th>Size (GB)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>500 M</td>
<td>1</td>
<td>9.4</td>
<td>2012</td>
</tr>
<tr>
<td>Medium</td>
<td>995 M</td>
<td>2</td>
<td>18.7</td>
<td>2012 - 2013</td>
</tr>
<tr>
<td>Large</td>
<td>2020 M</td>
<td>4</td>
<td>37.9</td>
<td>2013 - 2015</td>
</tr>
</tbody>
</table>
Benchmark design

- 4 combinations
# Results Loading

## Sand Engine

<table>
<thead>
<tr>
<th>Approach</th>
<th>Time (s)</th>
<th>Size (MB)</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>conversion</td>
<td>Load heap</td>
<td>Load IOT</td>
</tr>
<tr>
<td>xy - S</td>
<td>105.43</td>
<td>11.79</td>
<td>13.60</td>
</tr>
<tr>
<td>xy - M</td>
<td>145.14</td>
<td>16.56</td>
<td>49.65</td>
</tr>
<tr>
<td>xy - L</td>
<td>167.75</td>
<td>19.72</td>
<td>78.00</td>
</tr>
<tr>
<td>xyz - S</td>
<td>352.37</td>
<td>9.91</td>
<td>10.5</td>
</tr>
<tr>
<td>xyz - M</td>
<td>498.79</td>
<td>14.24</td>
<td>34.07</td>
</tr>
<tr>
<td>xyz - L</td>
<td>590.00</td>
<td>16.77</td>
<td>61.71</td>
</tr>
<tr>
<td>xyt - S</td>
<td>349.68</td>
<td>11.79</td>
<td>13.09</td>
</tr>
<tr>
<td>xyt - M</td>
<td>492.29</td>
<td>16.56</td>
<td>40.39</td>
</tr>
<tr>
<td>xyt - L</td>
<td>594.10</td>
<td>19.72</td>
<td>74.11</td>
</tr>
<tr>
<td>xyzt - S</td>
<td>435.48</td>
<td>11.79</td>
<td>10.78</td>
</tr>
<tr>
<td>xyzt - M</td>
<td>604.27</td>
<td>16.56</td>
<td>33.21</td>
</tr>
<tr>
<td>xyzt - L</td>
<td>722.08</td>
<td>19.72</td>
<td>57.96</td>
</tr>
</tbody>
</table>
Results Loading

- The **SFC conversion** is the most expensive phase.

- Adding one more dimension in the key decreases the performance of the conversion.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Time (s)</th>
<th>Load heap</th>
<th>Load IOT</th>
<th>Size (MB)</th>
<th>Points Heap</th>
<th>Points IOT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>conversion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xy - S</td>
<td>105.43</td>
<td>11.79</td>
<td>13.60</td>
<td>471</td>
<td>18,147,709</td>
<td>18,147,709</td>
</tr>
<tr>
<td>xy - M</td>
<td>145.14</td>
<td>16.56</td>
<td>49.65</td>
<td>1130</td>
<td>25,561,106</td>
<td>43,708,815</td>
</tr>
<tr>
<td>xy - L</td>
<td>167.75</td>
<td>19.72</td>
<td>78.00</td>
<td>1897</td>
<td>30,205,111</td>
<td>73,913,926</td>
</tr>
<tr>
<td>xyz - S</td>
<td>352.37</td>
<td>9.91</td>
<td>10.5</td>
<td>368</td>
<td>18,147,709</td>
<td>18,147,709</td>
</tr>
<tr>
<td>xyz - M</td>
<td>498.79</td>
<td>14.24</td>
<td>34.07</td>
<td>885</td>
<td>25,561,106</td>
<td>43,708,815</td>
</tr>
<tr>
<td>xyz - L</td>
<td>590.00</td>
<td>16.77</td>
<td>61.71</td>
<td>1495</td>
<td>30,205,111</td>
<td>73,913,926</td>
</tr>
<tr>
<td>xyt - S</td>
<td>349.68</td>
<td>11.79</td>
<td>13.09</td>
<td>471</td>
<td>18,147,709</td>
<td>18,147,709</td>
</tr>
<tr>
<td>xyt - M</td>
<td>492.29</td>
<td>16.56</td>
<td>40.39</td>
<td>1130</td>
<td>25,561,106</td>
<td>43,708,815</td>
</tr>
<tr>
<td>xyt - L</td>
<td>594.10</td>
<td>19.72</td>
<td>74.11</td>
<td>1897</td>
<td>30,205,111</td>
<td>73,913,926</td>
</tr>
<tr>
<td>xyzt - S</td>
<td>435.48</td>
<td>11.79</td>
<td>10.78</td>
<td>386</td>
<td>18,147,709</td>
<td>18,147,709</td>
</tr>
<tr>
<td>xyzt - M</td>
<td>604.27</td>
<td>16.56</td>
<td>33.21</td>
<td>927</td>
<td>25,561,106</td>
<td>43,708,815</td>
</tr>
<tr>
<td>xyzt - L</td>
<td>722.08</td>
<td>19.72</td>
<td>57.96</td>
<td>1566</td>
<td>30,205,111</td>
<td>73,913,926</td>
</tr>
</tbody>
</table>
Results Loading

- **Loading into the heap** table is not affected by the benchmark case used.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Time (s)</th>
<th>Size (MB)</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>conversion</td>
<td>Load heap</td>
<td>Load IOT</td>
</tr>
<tr>
<td>xy - S</td>
<td>105.43</td>
<td>11.79</td>
<td>13.60</td>
</tr>
<tr>
<td>xy - M</td>
<td>145.14</td>
<td>16.56</td>
<td>49.65</td>
</tr>
<tr>
<td>xy - L</td>
<td>167.75</td>
<td>19.72</td>
<td>78.00</td>
</tr>
<tr>
<td>xyz - S</td>
<td>352.37</td>
<td>9.91</td>
<td>10.5</td>
</tr>
<tr>
<td>xyz - M</td>
<td>498.79</td>
<td>14.24</td>
<td>34.07</td>
</tr>
<tr>
<td>xyz - L</td>
<td>590.00</td>
<td>16.77</td>
<td>61.71</td>
</tr>
<tr>
<td>xyt - S</td>
<td>349.68</td>
<td>11.79</td>
<td>13.09</td>
</tr>
<tr>
<td>xyt - M</td>
<td>492.29</td>
<td>16.56</td>
<td>40.39</td>
</tr>
<tr>
<td>xyt - L</td>
<td>594.10</td>
<td>19.72</td>
<td>74.11</td>
</tr>
<tr>
<td>xyzt - S</td>
<td>435.48</td>
<td>11.79</td>
<td>10.78</td>
</tr>
<tr>
<td>xyzt - M</td>
<td>604.27</td>
<td>16.56</td>
<td>33.21</td>
</tr>
<tr>
<td>xyzt - L</td>
<td>722.08</td>
<td>19.72</td>
<td>57.96</td>
</tr>
</tbody>
</table>
Results Loading

- The **creation of the IOT** is dependent only on the treatment of $z$ used.
- The IOT is created faster when treating $z$ as part of the key.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Time (s)</th>
<th>Load IO</th>
<th>Size (MB)</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>conversion</td>
<td>heap</td>
<td>IOT</td>
<td>Heap</td>
</tr>
<tr>
<td>xy - S</td>
<td>105.43</td>
<td>11.70</td>
<td>13.60</td>
<td>471</td>
</tr>
<tr>
<td>xy - M</td>
<td>145.14</td>
<td>16.50</td>
<td>49.65</td>
<td>1130</td>
</tr>
<tr>
<td>xy - L</td>
<td>167.75</td>
<td>19.72</td>
<td>78.00</td>
<td>1897</td>
</tr>
<tr>
<td>xyz - S</td>
<td>352.37</td>
<td>9.91</td>
<td>10.5</td>
<td>368</td>
</tr>
<tr>
<td>xyz - M</td>
<td>498.79</td>
<td>14.24</td>
<td>34.07</td>
<td>885</td>
</tr>
<tr>
<td>xyz - L</td>
<td>590.00</td>
<td>16.72</td>
<td>61.71</td>
<td>1495</td>
</tr>
<tr>
<td>xyt - S</td>
<td>349.68</td>
<td>11.75</td>
<td>13.09</td>
<td>471</td>
</tr>
<tr>
<td>xyt - M</td>
<td>492.29</td>
<td>16.50</td>
<td>40.39</td>
<td>1130</td>
</tr>
<tr>
<td>xyt - L</td>
<td>594.10</td>
<td>19.72</td>
<td>74.11</td>
<td>1897</td>
</tr>
<tr>
<td>xyzt - S</td>
<td>435.48</td>
<td>11.75</td>
<td>10.78</td>
<td>386</td>
</tr>
<tr>
<td>xyzt - M</td>
<td>604.27</td>
<td>16.50</td>
<td>33.21</td>
<td>927</td>
</tr>
<tr>
<td>xyzt - L</td>
<td>722.08</td>
<td>19.72</td>
<td>57.96</td>
<td>1566</td>
</tr>
</tbody>
</table>
Results Loading

- The **storage requirements** are affected only by the treatment of z.

- Treating z as an attribute increases the storage.
Results Querying

- Test the scalability of the queries

- Focus on the fetching time of the filter step that directly uses the structure. The rest of the steps can be improved in performance and are not analysed.
Results Querying

Space – time queries

![Graphs showing performance metrics for different datasets and query types.](image)
Results Querying

Space only queries
Results Querying

Time only queries
Conclusion & Future work
Conclusions

• Designed and executed a benchmark for dynamic point clouds

• Two integrations of space and time and, two treatments of $z$

• Integrated approach presented better query response times, compared to non-integrated for the specific use case (both treatments of $z$ possible)

• Key aspect of the implementation: Index Organised Table
Future work

- Native database functionality (encoding, decoding, range generation)
- Investigate a different SFC
- Investigate parallel processing
- Up-scaled benchmark of trillion points
- Investigate the generation of blocks: compression
Thank you for your attention!
References
